VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Rozšíření Jabber klienta o GIS funkce
Geographic Extension of Jabber Client

2013 **Bc. Lukáš Kohut**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student:	**Bc. Lukáš Kohut**
Studijní program:	N2647 Informační a komunikační technologie
Studijní obor:	2612T025 Informatika a výpočetní technika
Téma:	Rozšíření Jabber klienta o GIS funkce Geographic Extension of Jabber Client

Zásady pro vypracování:

Cílem diplomové práce je implementovat Jabber klienta s GIS rozšířením. Klient by využíval Extensible Messaging and Presence Protocol (XMPP), otevřený protokol na bázi XML. Úkolem diplomové práce by bylo navrhnout rozšíření, umožňující uživatelům lokalizovat klienty sítě Jabber. Výsledné řešení by mělo umožňovat zobrazení účastníků na mapovém podkladu a provádět jednoduché operace nad prostorovými dat, např. vyhledání účastníku v definovaném okruhu.

1. Protokol XMPP a jeho rozšíření.
2. Implementace Jabber klienta s GIS rozšířením (případně rozšířit již existující implementaci).
3. Návrh a testování doplňkových služeb (lokalizace, vyhledávání, navigace atd.).
4. Testování a shrnutí řešení.

Seznam doporučené odborné literatury:

Peter Saint-Andre , XMPP: The Definitive Guide: Building Real-Time Applications with Jabber Technologies, O'Reilly Media; 1 edition, 2009, ISBN 978-0596521264
D.J. Adams, Programming Jabber: Extending XML Messaging, O'Reilly Media; 1st edition, 2002, ISBN 978-0596002022
David Flanagan, Java In A Nutshell, O'Reilly Media; Fifth Edition edition, 2005, ISBN 978-0596007737

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr.Ing. Michal Krumnikl**

Datum zadání: 18.11.2011
Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne 31. 7. 2013 v Ostravě

.....

Bc. Lukáš Kohut

Poděkování

Rád bych poděkoval české komunitě Jabber a to zejména Martinu Tomašíkovi za konzultace k problematice XMPP. Dále Mgr. Ing. Michalu Krumniklovi za odbornou pomoc a konzultaci při vytváření této diplomové práce. A v neposlední řadě bych chtěl poděkovat rodině za trpělivost a podporu během celého studia.

Abstrakt

Diplomová práce se zabývá studiem XMPP protokolu a jeho využitím pro rychlou výměnu zpráv mezi uživateli. Prezentuje v dnešní době populární spojení rychlé komunikace, sdílení polohy s uživateli a využití mobilního zařízení.

Velká část diplomové práce je věnovaná popisu standardu XMPP protokolu, jeho využití v Jabber síti a charakteristiky operačního systému Windows Phone 7. Neméně důležitá část práce popisuje analýzu a implementaci mobilní aplikace postavené na standardu XMPP protokolu a operačního systému Windows Phone 7, která slouží pro výměnu textových zpráv a sdílení polohy mezi uživateli.

Klíčová slova

XMPP protokol, Jabber, Instant messaging, Windows Phone.

Abstract

This thesis applies the XMPP protocol and it's use for the rapid exchange of messages between users. Nowadays that presents a popular speed connection communication, sharing locations of users and usage of a mobile devices.

A large part of the thesis is focused for the description of the standard XMPP protocol, it's usage in the Jabber network and the characteristics of the operating system Windows Phone 7. Let's say an equally important stuff, we describe is the analysis and implementation of mobile applications has built on standard XMPP protocol and operating system Microsoft Windows Phone 7, which is has been used to exchange a text messages and as well as a location sharing between users.

Key words

XMPP protocol, Jabber, Instant messaging, Windows Phone.

Seznam použitých zkratek

XMPP Extensible Messaging and Presence Protocol

IM Instant Messaging

XML Extensible Markup Language

GIS Geographic information systém

RFC Request for comments

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

V dnešní uspěchané době, kdy lidé spolu tráví čím dál méně času, se stává nedílnou součásti života tzv. “Instant messaging”. Jedná se o označení, které se v angličtině vžilo pro rychlý způsob komunikace pomocí textových zpráv, posílaných mezi uživateli. Technologie nám dnes nabízí tento způsob komunikace jak na počítači, tak na mobilním zařízeni, díky neustálému přístupu na internet. Trendem poslední doby je spojení této komunikace s GIS (Geografickým informačním systémem). Geografické údaje posouvá komunikaci mezi uživateli do další dimenze a umožňuje mezi uživateli nejenom komunikovat, ale také umožnit sdílet aktuální geografickou polohu.

1.1. Vymezení cílů diplomové práce

Cílem diplomové práce je implementovat Jabber klienta s GIS rozšířením. Klient by využíval Extensible Messaging and Presence Protocol (XMPP), otevřený protokol na bázi XML. Úkolem diplomové práce je navrhnout rozšíření, umožňující uživatelům lokalizovat klienty sítě Jabber. Výsledné řešení by mělo umožňovat zobrazení účastníků na mapovém podkladu a provádět jednoduché operace nad prostorovými daty, např. vyhledání účastníku v definovaném okruhu.

1. Protokol XMPP a jeho rozšíření.
2. Implementace Jabber klienta s GIS rozšířením (případně rozšířit již existující implementaci).
3. Návrh a testování doplňkových služeb (lokalizace, vyhledávání, atd.).
4. Testování a shrnutí řešení.

1.2. Požadavky na implementovaný software

Software by měl splňovat všechny požadavky stanovené v cílech diplomové práce. Pro přidanou hodnotu implementovaného softwaru bylo rozhodnuto o kontaktování české komunity vývoje sítě jabber, která se zasloužila o distribuci klienta s názvem Jabbim. Česká komunita doposud disponovala desktopovým klientem a klientem pro mobilní operační systém Android. Na základě konzultací a potřeb české komunity se vykrystalizovaly dvě cesty vývoje.

1. Rozšíření stávajícího klienta o GIS prvky.
2. Implementace nového klienta pro operační systém Windows Phone 7.

Pro rozšíření portfolia české komunity klientů byla zvolena cesta naprogramování nového klienta pro operační systém Windows Phone 7. Cesta vývoje pro operační systém Windows Phone 7 byla také zvolena z důvodu malé konkurence na poli jabber klientů na této platformě a její distribuci pomocí obchodu s aplikacemi Marketplace od společnosti Microsoft.

Požadované vlastnosti softwaru:

- Klient poběží v operačním systému Windows Phone 7.
- Klient bude umožňovat komunikaci přes XMPP protokol.
- XMPP protokol musí umět podporovat komunikaci přes různé servery.
- GIS bude umožňovat zobrazovat uživatele sdílející polohu a umožňovat zobrazení na mapě.
- Pomocí GIS bude možno filtrovat uživatele na základě různých vzdáleností.
- XMPP knihovna by měla splňovat standarty internetu RFC 3920, RFC 3921.
- Klient musí dodržovat standardy požadované společnosti Microsoft pro schválení v Marketplace.

1.3. Existující řešení Jabber klientů ve Windows Phone

Mapováním trhu aplikací pro Windows Phone, bylo zjištěno, že existují dvě plnohodnotné aplikace, které využívají XMPP protokol pro komunikaci mezi uživateli. Tyto aplikace můžeme zařadit do skupiny existujících řešení Jabber klientů. Jedná se tedy o přímou konkurenci navrhované aplikace. Obě aplikace jsou placené, pokud je chceme plnohodnotně využívat. V případě porovnání těchto aplikací s navrhovanou aplikací je hlavní výhoda rozšíření o geografické prvky. Žádná z konkurenčních aplikací nenabízí možnost sdílení a zpracování polohy uživatelů.

1.3.1. JabberTalk

JabberTalk je aplikace používající komunikační protokol XMPP pro zasíláním zpráv a komunikaci mezi uživateli. Jedná se o jedinou aplikaci přístupnou přes český Marketplace, která

používá pro komunikaci pouze XMPP protokol. Aplikace je nabízená ve dvou verzích. První verze je zpoplatněná a neobsahuje žádné omezení. Druhá verze se nabízí v marketplace zadarmo, ale oproti placené verzi je omezená pouze na prvních padesát přihlášení do aplikace.

1.3.2. IM +

IM + jedná se o aplikace pro zasílání zpráv a komunikaci mezi uživateli. Aplikace pro komunikaci využívá nejenom XMPP protokol, ale také jiné komunikační protokoly. Aplikace je dostupná na českém Marketplace. Uživatel může zvolit placenou nebo neplacenou verzi. Placená verze aplikace umožňuje přijímat notifikační zprávy, které umožní komunikaci i tehdy, kdy je aplikace vypnuta.

2. XMPP Protokol

Jelikož se jedná o rozsáhlý standart protokolu budu se snažit v následující kapitole vypíchnout všechny základní a důležité prvky protokolu.

2.1. Úvod

Extensible Messaging and Presence Protocol (XMPP) , neboli „rozšiřitelný protokol pro posílání zpráv a zjištění stavu“. Původně protokol vznikl pro IM (instant messaging) síť s názvem Jabber, kterou v roce 1998 vytvořil Jeremie Miller.[1] Brzy se však ukázalo, že kromě IM může být s výhodou použit pro vzájemnou komunikaci programů nebo pro ovládání různých automatizovaných služeb. Použitelnost protokolu zajistilo zařazení jakožto standart Internetu do RFC[1] dokumentů.

RFC standardy pro definici XMPP jádra a IM komunikace:

- RFC 3920 – „XMPP: Core“
- RFC 3291 – „XMPP: Instant Messaging and Presence“

XMPP je postaveno, tak aby mohlo dojít k rozšíření protokolu o chování entit pro různá řešení. Všechny standardizovaná rozšíření jsou definovaná v dokumentech zvaných XEP (XMPP Extension Protocol). Všechny dokumenty definující nějaká rozšíření XMPP protokolu jsou číslovány ve format XEP-0000. V této práci se bude podrobně věnovat pouze jednomu rozšíření, které popisuje uživatelskou polohu.

- XEP-0080 User location.

Protokol XMPP běží na aplikační vrstvě síťové architektury ISO/OSI[2]. Je určen pro doručování zpráv v reálném čase. Základním stavebním kamenem protokolu je XML. Veškerá komunikace mezi koncovými body zajišťovaná protokolem XMPP probíhá pomocí zpráv odpovídajících XML syntaxi.[2]

[1] Request for comments je označení pro standarty a dokumenty popisující internetové protokoly.
[2] Referenční model ISO/OSI popisuje komunikaci v počítačových sítí pomocí vrstevnatého modelu.

2.2. Stručný úvod do XML

Extensible Markup Language (XML) je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C[3]. Je zjednodušenou podobou staršího jazyka SGML.[6] Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a různé typy dat.

V současnosti se jazyk XML používá v mnoha aplikacích a technologiích. Hlavně se používá jako formát pro ukládání dat nebo formát pro výměnu informací. Nejčastěji se využívá v aplikacích jako formát mnoha konfiguračních souborů, či jako formát pro serializaci objektů. Důležitým prvkem je XML také v architektuře SOAP – protokol pro komunikaci mezi webovými službami. Dále se XML využívá jako metajazyk pro definici dalších jazyků. Mezi nejznámější jazyky postavené na XML patří XHTML, nástupce jazyka HTML, který plně přebírá jeho sémantiku, ale syntaxe je přizpůsobena XML.

XML patří mezi značkovací jazyky (markup languages). Důležité části dokumentu se označují pomocí značek. V terminologii XML se jednotlivým označeným částem dokumentu říká elementy. Elementy do sebe mohou být navzájem vnořené a tím dle potřeby zachycovat strukturu informací uložených v dokumentu. Element je tvořen třemi částmi. První část je otevírací tag, za ním následuje tělo elementu a celý element je zakončen uzavíracím tagem. Otevírací tag obsahuje povinnou položku název tagu a nepovinné rozšiřující položky tzv. atributy. Atribut se skládá z povinných položek a to klíč a hodnoty. Tělo elementu může obsahovat jiné elementy nebo obsah tvořený textem. Uzavírací element obsahuje název tagu, který odpovídá otevíracímu tagu.

Dokument považován za správně strukturovaný musí splňovat následující vlastnosti:

- Obsahuje právě jeden kořenový element (root).
- Všechny hodnoty atributů musí být ohraničeny uvozovkami - jednoduchými (‘) nebo dvojtými (“).
- Elementy mohou být vnořeny, ale nemohou se překrývat.

[3] World Wide Web Consortium je mezinárodní konsorcium, jehož členové společně s veřejnosti vyvíjejí webové standardy pro World Wide Web www.w3c.org.

Příklad XML:

```
<Článek autor=”Pepa Nový”>
      <Nadpis>Pokusný nadpis</Nadpis>
      <Odstavec>První odstavec</Odstavec>
      <Odstavec>Druhý odstavec</Odstavec>
      <Odstavec>Třetí odstavec</Odstavec>
</Článek>
```

2.3. Základní principy komunikace protokolu

Protokol XMPP je postavený na dvou entitách klient a server. V podstatě, ale protokol nezná klasický pojem server ani pojem klient. Entity rozlišujeme pouze pro jejich zodpovědnost a povinnost k sobě navzájem. Klient a server chápeme jako dvě aplikace, které mezi sebou komunikují pomocí XMPP protokolu.

Serverová aplikace je zodpovědná za:

1. Správu spojení a session mezi entitami.
2. Autorizace entit (klientů, serverů).
3. Přeposílání XML zpráv mezi entitami.

Mimo základní zodpovědnosti XMPP serveru patří také další povinnosti, jako jsou uchovávaní nedoručených zpráv pro nedosažitelné uživatele daného serveru nebo seznam kontaktů uživatele tzv. roaster[4].[4] Serverová část je mnohem sofistikovanější oproti klientské aplikaci. Tato práce je především věnovaná klientské části, proto se dále budeme věnovat pouze popisu této části.

Klient se vždy identifikuje vůči serveru pomocí unikátního identifikátoru s názvem JID – Jabber ID, který je jedinečný v celé síti XMPP/Jabber a slouží jako uživatelské jméno uživatele.[4] JID je strukturován podobně jako e-mailová adresa. Skládá se ze tří částí:

1. Uživatelské jméno – identifikace uživatele na konkrétním serveru.
2. Doména serveru – identifikace serveru.
3. Zdroj – identifikace klientské aplikace.

[4] Roaster je v terminologii XMPP protokolu nazýván seznam kontaktů jednoho uživatele.

Výsledný formát je *username@domain.tld/resource*. Díky tomuto schématu není potřeba zavádět centrální server, který obsahuje všechny uživatelské účty sítě, protože uživatel se vždy autorizuje vůči jednomu serveru. Dokonce toto schéma zajišťuje, že je možné se přihlásit na XMPP server z různých klientů najednou. Zajišťuje to identifikátor – tzv. *resource*, který jednoznačně určuje jednotlivé klienty, připojené k serveru pomocí daného JID. V případě, že klient se identifikuje bez části resource, server generuje vlastní náhodný resource pro identifikaci a jednoznačné určení klienta.

Klient komunikuje vždy přímo se serverem, který se stará o další přeposílání zpráv. Také klient může využívat služeb serveru. Každý server nabízí vlastní rozšířené služby, o kterých informuje klienta.

2.4. Funkční shrnutí protokolu

Účelem protokolu XMPP je umožnit mezi entitami výměnu relativně malého počtu strukturovaných zpráv, v našem případě definované XML jazykem. Těmto strukturovaným zprávám v terminologii XMPP říkáme „XML Stanza“.[3] Tyto zprávy posíláme mezi entitami pomocí transportního protokolu TCP[5].

Proces spojení XMPP protokolu probíhá v následujících krocích:

1. Otevření transportního spojení TCP.
2. Otevření XML streamu.
3. Sjednání TLS šifrování, pokud je možno.
4. Autentizace pomocí SASL mechanismu.
5. Výměna informací o funkcích serveru.
6. Výměna XML stanza.
7. Ukončení XML streamu.
8. Ukončení spojení TCP.

Z procesu spojení je vidět, že celá komunikace mezi entitami je zahájena TCP spojením a poté otevřením XML streamu. Bezpečnost komunikace zabezpečuje technologie TLS pro šif-

[5] Transmission Control Protocol je protokol představující transportní vrstvu.

rování komunikace a pro autentizaci technologie SASL[6], která umožňuje několik možností přenosu uživatelského jména a hesla. Po úspěšné autentizaci a autorizaci uživatele, server informuje klienta o dostupných službách. Následuje výměna zpráv pomocí XML stanz a po provedení veškeré výměně zpráv se komunikace ukončuje, uzavřením XML streamu a TCP spojení.

2.5. XML stream

Jedná se o kontejner pro výměnu XML stanz mezi entitami v síti. V terminologii XML se jedná o kořenový element. Proto celá komunikace je ve výsledku zaznamenána v jednom XML dokumentu, který v průběhu celé komunikace roste. Dokument je tedy definován kořenovým elementem <stream/>.[3]

Příklad otevření XML streamu:

```
<?xml version='1.0'?>
<stream:stream
        from='user@domain.tld'
        to='domain.tld'
        version='1.0'
        xml:lang='en'
        xmlns='jabber:client'
            xmlns:stream='http://etherx.jabber.org/streams'>
```

Příklad ukončení XML streamu:

```
</stream:stream>
```

Ukončení XML streamu se provádí před uzavřením spojení. Není to však jediná možnost, kdy se spojení ukončí uzavřením XML streamu. Další důvody uzavření streamu jsou:

- Neúspěšný pokus o navázání TLS.
- Po neúspěšné autentizaci pomocí SASL.
- V případě chyby při vytváření spojení.
- V případě závažné chyby.

[6] SASL – Simply Authentication and Security Layer je metoda ověřování v protokolech klient/server.

V seznamu důvodu předčasného ukončení XML streamu se vyskytuje důvod ukončení „V případě závažné chyby“. Závažná chyba se identifikuje na straně serveru pomocí aplikační logiky. Server kontaktuje klienta o závažné chybě pomocí odeslání elementu <stream:error>, který obsahuje popis dané chyby. To má za následek ukončení jak XML streamu, tak TCP spojení mezi entitami.

2.6. XML stanza

Základní stavební kámen protokolu. Definuje strukturu jednotlivých zpráv, které se posílají mezi entitami. Rozlišujeme tři základní typy zpráv stanza:

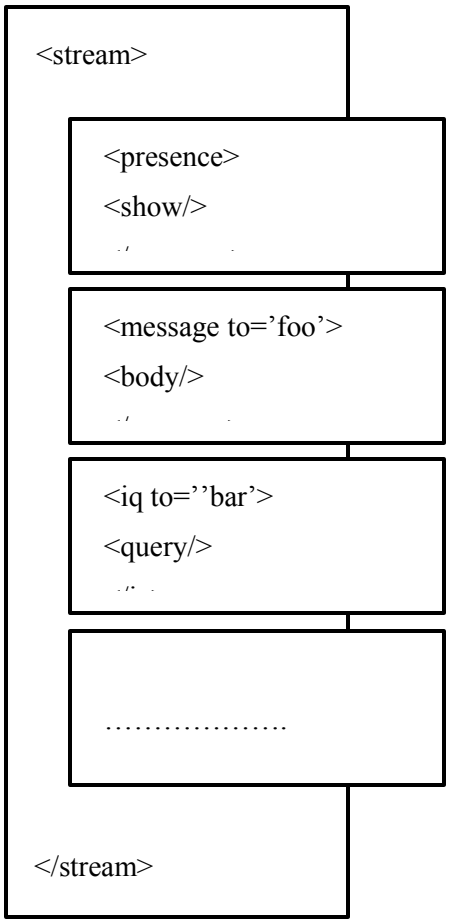
1. <message/>
2. <presence/>
3. <iq/>

Všechny tři typy zpráv definované pomocí XML elementů popisují dva základní namespace „jabber:client“ a „jabber:server“. Každá zpráva obsahuje povinné atributy.

Společné povinné atributy:

- To - Specifikuje adresáta zprávy. V serverové namespace se jedná o povinnou položku a obsahuje vždy JID adresáta. V klientské namespace může být atribut prázdný a znamená to, že server má považovat zprávu za broadcast a rozeslat všem kontaktům.
- From – Specifikuje odesílatele zprávy pomocí JID.
- Id – Identifikátor zprávy.
- Type – Typ zprávy.
- Xml:lang – Určuje v jakém jazyce jsou data psaná.

Příklad komunikace klienta ve vytvořeném XML streamu:



Obrázek 1 : XML stream.

2.6.1. Message stanza

Jedná se o zprávu, která se předává z klienta na server a obvykle nevyžaduje potvrzení. Používá se pro rychlé předávání zpráv. Existuje pět typů zpráv rozlišené podle důvodu jejich vzniku:

- Normal – Používá se pro obyčejnou komunikaci mezi uživateli, kdy není vyžadována historie zpráv.
- Chat – Označuje jednu zprávu v konverzaci mezi uživateli.
- Groupchat – Používá se při komunikaci mezi více uživateli.
- Error – Označuje zprávu popisující chybu.

- Headline – Zpráva, která slouží k informování uživatele.

Příklad nejčastěji používané zprávy:

```
<message from='juliet@im.example.com/balcony'
        id='ju2ba41c'
        to='romeo@example.net'
        type='chat'
        xml:lang='en'>
        <body>Art thou not Romeo, and a Montague?</body>
</message>
```

2.6.2. Presence stanza

Zpráva informující o stavu entity v síti. Stav určuje, zda entita je přístupná ke komunikaci či nikoliv. Protokol XMPP umožňuje tzv. presence subscription, který definuje schéma, zda dvě entity jsou autorizované k tomu, aby si navzájem mohli vyměnit aktuální stav.[4] Schéma popisuje čtyři stavy:

- None - Neexistuje autorizace.
- From, To – Jednosměrná autorizace.
- Both – Obousměrná autorizace.

Autorizace se ukládá u každého kontaktu. Každý kontakt je opět identifikován svým Jabber ID. Kontakty pro jednoho uživatele se ukládají na serveru i se stavy autorizace mezi entitami.

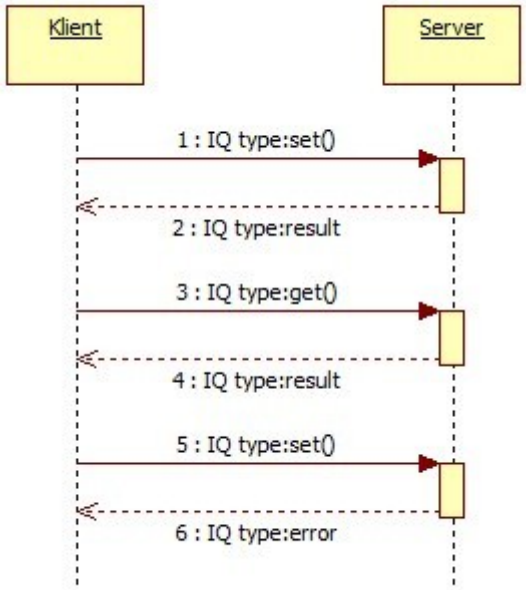
Presence stavy:

- On – Entita je on-line.
- Away – Entita je pryč.
- Chat – Entita je k dispozici pro komunikaci.
- Dnd – Entita nechce být rušena.
- Xa – Entita je delší dobu pryč.
- Off – Entita je off-line.

2.6.3. IQ stanza

Posledním typem zprávy je IQ (Info/Query) stanza.[3] Zpráva je určena pro obecnou výměnu dat mezi entitami. Reprezentuje model otázka-odpověď, což znamená, že když entita odesílá tento typ zprávy, tak bude očekávat i odpověď od adresáta. Každá otázka a odpověď je definována svým unikátním Id, tak aby se mohli spárovat. Existují čtyři typy IQ stanza:

- Get – Žádost o data.
- Set – Poskytnutí dat nebo žádost o provedení určité akce.
- Result – Odpověd na zprávu get nebo set.
- Error – Informace o chybě způsobenou předchozím požadavkem.



Obrázek 2: Diagram znázorňující průběh výměny IQ stanza zpráv.

Ze schématu komunikace jde vidět, že zprávy typu get a set musí mít vždy odpověď tvořenou zprávou typu result nebo v chybovém stavu zprávou typu error.

Příklad nastavení resource pomocí zprávy IQ:

Set

```
<iq id='1' type:'set' xml:lang='en'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
        <resource>WPJabbim</resource>
     </bind>
</iq>
```

Result

```
<iq id='1' type:'result'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
        <jid>example@domain.dtl/WPJabbim</jid>
     </bind>
</iq>
```

2.7. Rozšíření protokolu XMPP

Na začátku kapitoly jsem popisoval, že existuje různé standardizované rozšíření protokolu definované pomocí XEP dokumentů. Pro tyto rozšíření existuje vždy podrobná dokumentace, která obsahuje definici použití. V mnoha případech si s těmito rozšířeními nemusíme vystačit. Proto lze protokol modifikovat pro vlastní specifické použití. Tato úprava se provádí úpravou existujících elementů.

2.7.1. Úprava existujících elementů

Princip spočívá ve využití existujících elementů, to znamená využití jedné z XML stanz zpráv. Rozšíření definujeme jak subelement zprávy obsahující vlastní namespace, které rozšíření definuje. XMPP protokol je tolerantní na různý obsah stanz.

Příklad vlastního rozšíření protokolu:

```
<message to='romeo@example.net'
        xml:lang='en'>
        <body>Art thou not Romeo, and a Montague?</body>
        <extensions xmls='http://jabber.org/protocol/customextensions'>
                <custom>Vlatní hodnota</custom>
        </extensions>
</message>
```

Příklad znázorňuje jak si rozšířit XML stanz zprávu o vlastní strukturovaná data. V případě použití vlastního rozšíření, ale nastává situace, že příjemce zprávy nemusí vždy znát strukturu rozšíření. Entita přijímající rozšiřující zprávu identifikuje známé prvky a ostatní neznámé ignoruje, což předchází různým problémům.

3. Jabber síť

Kapitola popisuje postavení jednotlivých entit v síti a popisuje rozdíl mezi použití výrazu XMPP protokol a Jabber.

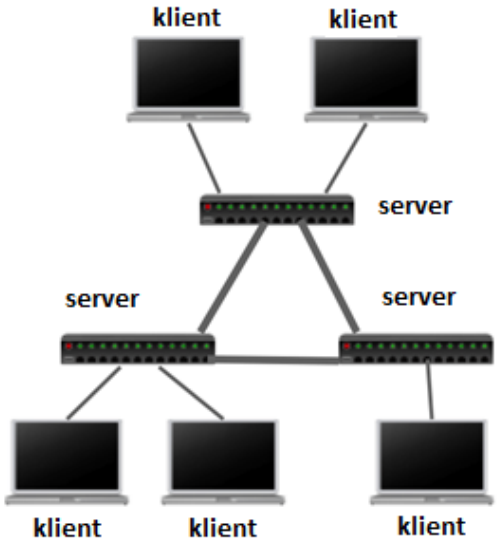
3.1. Charakteristika sítě Jabber

V předchozí kapitole jsme si popsali co je to XMPP protokol, ale jaký je význam mezi výrazem Jabber a XMPP? Jabber definuje instatnt messagingovou síť, která je založena na protokolu XMPP. Původně byla vytvořena síť Jabber a na základě této sítě bylo standardizované jádro a rozšíření IM pomocí RFC dokumentů.

3.2. Architektura sítě

Síť Jabber je založena na síťové architektuře klient-server a je decentralizovaná stejně jako e-mail. To znamená, že neexistuje žádný centrální server, který by udržoval informace o všech uživatelích. Na základě architektury vznikají dva typy komunikace:

- Klient – Server
- Server – Server



Obrázek 3: Architektura sítě Jabber.

Jelikož je sít decentralizovaná, může si každý zřídit vlastní server, přičemž bude komunikovat s uživateli na jiných serverech. Uživatel tak má svobodu volby serveru. Servery mohou nabízet různé služby a rozšíření protokolu, podle kterých uživatel může volit server.

Každý server je na sobě nezávislý a udržuje si vlastní uživatele. Server o ostatních serverech neví až do chvíle, kdy jeden z uživatelů chce komunikovat s uživatelem jiného serveru. Pokud server naváže spojení s jiným serverem za účelem komunikace, uloží si spojení s tímto serverem pro budoucí použití z důvodu šetření prostředků.

Komunikace mezi serverem a klientem používá spojení běžící na dvou výchozích portech. V případě spojení server-server je port 5269 a v případě spojení klient-server je port 5222.

3.3. Přeposílání zpráv

Už známe strukturu zpráv pro posílání v síti Jabber, ale neznáme princip přeposlání zprávy z jednoho klienta na jiného klienta. Komunikace mezi klienty začíná posláním zprávy od jednoho klienta vždy domovskému serveru, na kterém je uživatel registrován. Server provede analýzu zprávy a identifkuje její části. Na základě analýzy určí JID adresáta. JID umožní zjistit adresu cílového serveru, kam zašle celou zprávu. Cílový server zprávu přijme a na základě JID provede validaci zprávy podle uživatelem definovaných pravidel pro příjem zpráv. Při úspěšné validaci server přepošle zprávu adresátovi, v opačném případě celou zprávu server zahazuje. Nastane-li, že adresát nebude připojen k cílovému serveru, server podle svých specifikací buď zprávu uchová do doby, kdy se uživatel na server připojí nebo zprávu zahodí a pošle odesílateli informaci o nedoručení zprávy. Druhá varianta nastává v dnešní době už ojediněle, protože většina dnešních serverů zprávy uchovává pro pozdější doručení.

4. GIS

Kapitola zkráceně přiblíží, co pojem GIS znamená a kde všude se s tímto pojmem můžeme setkat.

4.1. Charakteristika GIS

Zkratka GIS se používá pro označení Geografického Informačního Systému. Zjednodušeně můžeme říct, že se jedná o informační systém rozšířený o metody, které využívají prostorovou analýzu geografických dat a tvorbu map.[7] Každý geografický informační systém se skládá z pěti základních složek:

1. **Hardware -** počítačové vybavení, které se používá pro zpracování geografických dat. Mezi vybavení zde patří i mobilní zařízení, které obsahují GPS přijímače pro zjišťování polohy.
2. **Software -** programové vybavení, které se používá pro prostorovou analýzu geografických dat.
3. **Data -** nejdůležitější část geografického informačního systému.
4. **Metody -** postupy podle kterých GIS pracuje a je používán.
5. **Lidé -** uživatelé používající GIS.

4.2. Vlastnosti geografických dat v GIS

Každá informace geografického prostoru má jasně definovanou svou prostorovou, atributovou a časovou informaci. Prostorová geografická informace popisuje polohu geografického objektu v krajinné sféře, nejčastěji tvořena pomocí zeměpisných souřadnic. Atributová geografická informace popisuje vlastnost daného objektu. Poslední geografickou informací je časová informace, která přidává do systému dynamickou vlastnost, například kdy byl objekt naposledy navštíven.

4.3. Prostorová data v analogové podobě

Jednotlivé objekty jsou v klasické analogové mapě reprezentovány pomocí následujících prvků:

- Bod – reprezentuje body, které nemají žádný rozměr nebo jsou příliš malé pro zaznamenání pomocí jiného prvku. Objekt má dimenzi 0 – nelze u něj měřit žádný rozměr.
- Linie – reprezentuje úzké objekty, jakou jsou silnice a řeky nebo objekty, které nemají definovanou šířku, jako jsou vrstevnice. Objekt má dimenzi 1- lze u něj měřit délku v jednom směru.
- Plocha – reprezentuji objekty, které jsou ohraničeny a uzavírají nějakou homogenní oblast, jako jsou vodní nádrže, lesy. Objekt má dimenzi 2 – lze jej měřit ve dvou směrech.

4.4. Prostorová data v digitální podobě

Digitální podoba pro prostorová data definuje dva modely reprezentace. Jedná se o modely vyjádření dat pomocí vektorového a rastrového formátu. Ve vektorovém formátu je poloha geografických objektů určena zápisem kartézských souřadnic x a y reprezentující body objektu. Vektorová data jsou přesná a vhodná pro reprezentaci celé mapy. Nevýhoda reprezentace geografických objektů vektorovým formátem je vysoká náročnost prostorových operací. V rastrovém formátu je základním nositelem informace tzv. pixel[7], který je umístěný v pravidelné síti řádků a sloupců. Rastrová data jsou méně přesná než vektorová data. Hlavní využití rastrového formátu je pro znázornění mapy jevů na určitém území. Mezi jevy, které takto můžeme reprezentovat, jsou teplota, množství srážek atd. Rastrový formát se také využívá při snímkování země, kdy se snímky ukládají jako rastrové obrázky.

[7] Pixel je označení pro nejmenší jednotku digitální rastrové grafiky.

5. Windows Phone

V kapitole se seznámíme s mobilním operačním systémem společnosti Microsoft s názvem Windows Phone. Ohlédneme se do historie vývoje mobilních operačních systémů nabízených společnosti Microsoft a nakonec zmapujeme verzi Windows Phone 7, která se stala platformou pro realizaci vlastního mobilního Jabber klienta.

5.1. Historie mobilních operačních systémů společnosti Microsoft

Společnost Microsoft má v oblasti vývoje operačních systémů pro mobilní zařízení bohatou minulost, která před více jak patnácti lety začala systémem Windows CE. Tento operační systém byl založen na kódu z Windows 95, kterému se mělo podobat i uživatelské rozhraní s kódovým označením WinPad. Windows CE nakonec nebyl uveden na trh jako spotřební výrobek pro uživatele, protože se od začátku potýkal s problémy výkonu a ovladači dotykového ovládání, ale na druhou stranu vývoj přinesl nové inovace, které se použili v budoucích zařízeních.

Významnou roli pro Microsoft hrálo až představení operačního systému Pocket PC. Původně byl uveden jako systém pro zařízení PDA, nicméně při propojení systému s mobilními telefony se dostalo na přejmenování systému názvem Windows Mobile, který započal dlouhou historii operačních systémů pod označením Windows Mobile. Rodina operačních systémů Windows Mobile byla známa pro svou otevřenost k vývojářům a použití ve firemní sféře. Byla postavena na jádru Windows CE. Vývojáři měli velké možnosti provádění úprav operačního systému. Poslední verzí v řadě Windows Mobile byla 6.5.3. Potom přišla ze strany Microsoftu revoluce v podobě nového operačního systému s označením Windows Phone 7. Nový operační systém byl představen v roce 2010. Revoluce zasáhla hlavně uživatelské rozhraní, které bylo úplně odlišné od přechozích verzí Windows Mobile a nepodobalo se ani uživatelským rozhraním systémů jiných výrobců jako je Apple či Google.

Název mobilního operačního systému	Verze Windows CE
Windows Mobile 6.5.3	5.2.23090
Windows Mobile 6.5	5.2.21234
Windows Mobile 6.1.4	5.2.20757
Windows Mobile 6.1	5.2.19202
Windows Mobile 6.0	5.2.x
Windows Mobile 5.0	5.1.x
Windows Mobile 2003 Second Edition	4.21
Windows Mobile 2003	4.20

Tabulka 1: Přehled mobilních operačních systémů a verzí Windows CE.

Tabulka zobrazuje přehled rodiny mobilních operačních systémů Windows Mobile a k nim konkrétní verzi jádra Windows CE.

5.2. Specifika Windows Phone 7

Příchodem operačního systému Windows Phone 7 se Microsoft pokusil o sjednocení jak softwarové stránky systémů ve smyslu designu aplikací, tak taky nastavení minimálních hardwarových požadavků výrobcům zařízení. Všichni výrobci musí splňovat minimální hardwarové požadavky pro to, aby na jejich přístrojích mohl běžet operační systém Windows Phone. Tato myšlenka je správná a s tímto rozhodnutím odpadají problémy s pomalostí systémů na různých přístrojích, které někdy Microsoft měl se systémem Windows Mobile. Sjednocení softwarové stránky aplikací zajišťuje centrálním místem pro distribuci aplikací třetích stran s názvem Marketplace, kde každá aplikace prochází podrobným testem, zda splňuje všechny požadavky stanovené Microsoftem pro publikaci aplikace. Tím zabraňuje šíření škodlivých aplikací mezi uživateli.

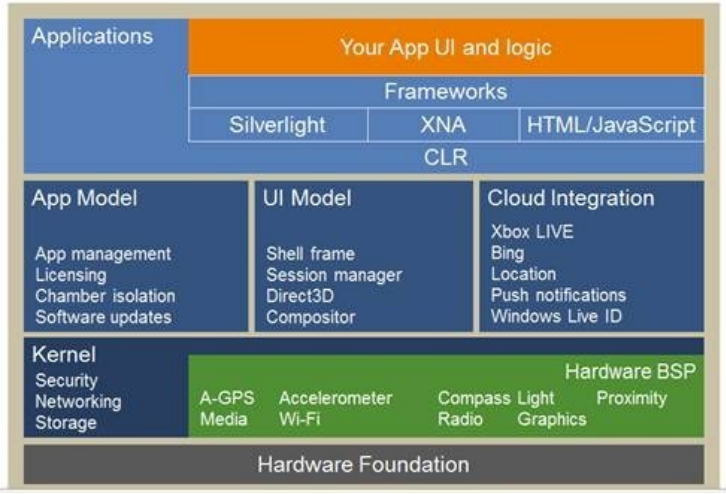
5.2.1. Hardwarové požadavky

Každý výrobce mobilních zařízení, na kterém běží mobilní operační systém Windows Phone 7, musí splnit následující minimální hardwarové specifikace:

- Rozlišení obrazovky – WVGA (480x800).
- Paměť – 256 MB RAM.
- Úložiště – 4 GB.
- CPU – architektura ARM v7.
- Dotykový displej – více dotykový, 4-bodový.
- Hardwarové tlačítka – back, start, search.
- Digitální fotoaparát – 5Mpx.
- GPS, Akcelerometr, Kompas, FM rádio a WiFi.

5.2.2. Architektura platformy WP7

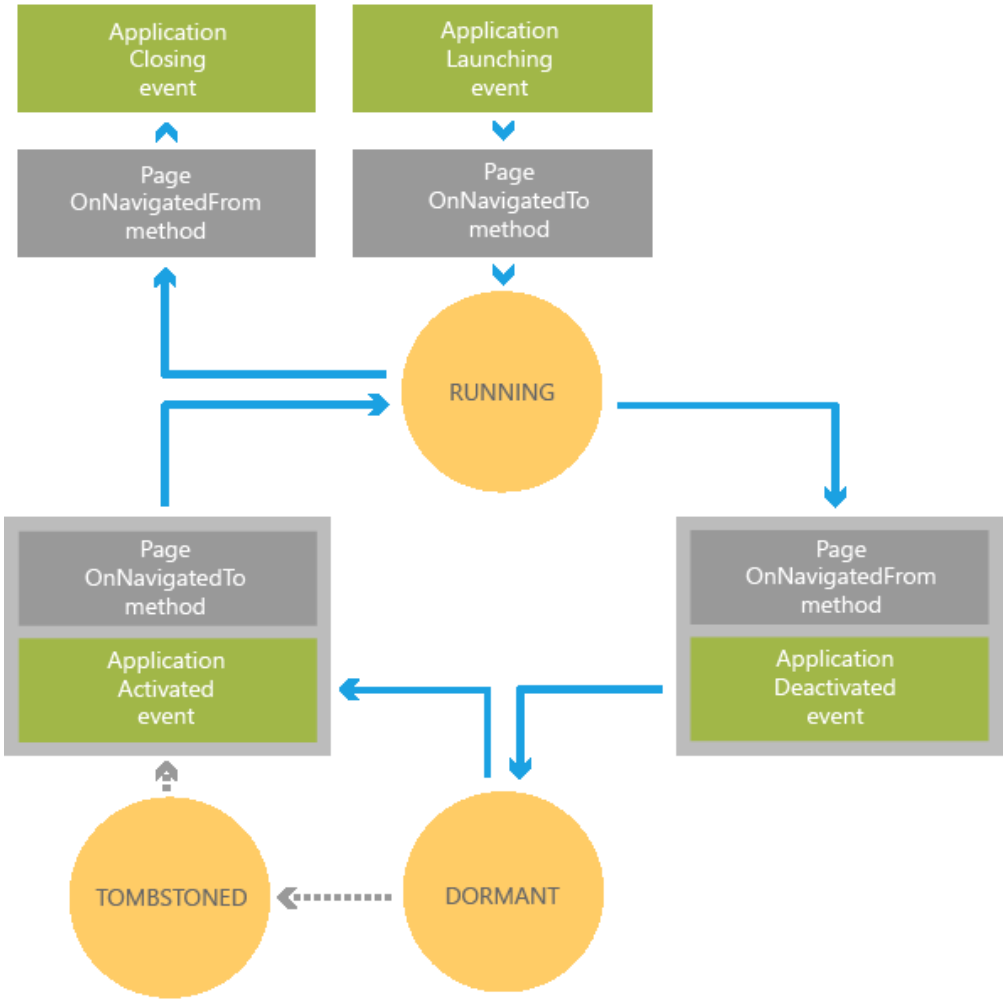
Architektura umožňuje podobně jako operační systém Android od společnosti Google provozovat operační systém na mobilních zařízeních od různých společností. Rozdíl od systému Android je v tom, že Microsoft pro zajištění konzistence vlastnosti zařízení vydefinoval minimální hardwarové nároky na přístroj. Pro použitý hardware poskytuje většinu ovladačů. Windows Phone využívá vícevrstvou architekturu a hardwarová akcelerace je zapouzdřena pomocí vrstvy DirectX nebo XNA. Architekturu popisuje diagram [Obrázek 4].



Obrázek 4: Architektura operačního systému Windows Phone 7.

5.2.3. Životní cyklus aplikace

Windows Phone je specifický tím, že na popředí běží v jednom okamžiku vždy jedna aplikace. To znamená, že v jeden okamžik běží jedna instance. Každá aplikace má svůj životní cyklus, ve kterém nabývá různých stavů. Stavy se mění podle vyvolaných akcí při přechodu mezi aplikacemi. Životní cyklus popisuje diagram [Obrázek 5].



Obrázek 5: Životní cyklus aplikace.

Nová instance aplikace se vytváří při spuštění aplikace pomocí ikony v seznamu instalovaných aplikací. Na základě vyvolané akce se provádějí následující události:

Launching

Po spuštění nové instance aplikace je aktivována událost Launching. Tato událost by měla provádět co nejméně logiky, vyvarovat se náročným úkolům, jako je práce se soubory nebo síťové operace. Všechna složitější logika by se měla vykonat na pozadí až po spuštění aplikace.

Running

Jakmile je aplikace spuštěna, přechází do stavu Running. Stav se nemění do chvíle, než uživatel aplikaci ukončí nebo se aplikace deaktivuje kvůli nečinnosti.

OnNavigatedFrom

Tato metoda je volána vždy, když uživatel přechází pryč z aktivní stránky na jinou stránku v rámci běžící aplikace.

Deactivated

Jedná se o událost, která je aktivována, když uživatel stiskne tlačítko Start nebo spustí jinou aplikaci. Deactivated je také aktivována, pokud aplikace spustí task typu Chooser[8]. V této události by se měly uložit všechny neuložené data aplikace, aby mohly být později obnoveny. Pro uložení dat nabízí Windows Phone speciální uložiště, které má strukturu slovníku a slouží pro ukládání stavů objektů pro pozdější použití.

Dormant

Když se uživatel dostane mimo aplikace, pokusí se operační systém převést aplikaci do režimu spánku. V tomto stavu se všechny běžící vlákna aplikace zastaví, ale aplikace zůstane neporušená v paměti. To znamená, že všechny objekty zůstanou v paměti aktivní. V případě, že se uživatel vrátí do aplikace, spustí se zpátky vlákna a pokračuje s objekty, které zůstaly v paměti. V případě, že operační systém vyhodnotí, že je potřeba uvolnit prostředky pro jiné aplikace, uvede spící aplikaci do stavu Tobstoned.

[8] Chooser je API, které dovoluje použití vestavěných funkcí pro uživatele, například pořízení fotografie fotoaparátem.

Tombstoned

Aplikace v tomto stavu byla ukončena. Informace o stavech aplikace, které jsou uloženy ve slovníku, jsou udržovány díky naplnění v události Deactivated. Informace o stavech se udržuje pro maximálně pět běžících aplikací. Pokud je tento limit překročen, další aktivace aplikace se spouští jako nová instance.

Activated

Událost Activated je volána tehdy, když se uživatel vrátí do aplikace ze stavu Dormant nebo Tombstoned. Systém provede kontrolu vlastnosti IsApplicationInstancePreserved, která určí, zda došlo k obnovení aplikace, která byla ve stavu Dormant nebo Tombstoned. Pokud vlastnost nabývá hodnoty True, jedná se o aplikaci, která byla ve stavu Dormant a tedy objekty jsou ještě v paměti a dojde k rychlému obnovení aplikace. V opačném případě aplikace byla ve stavu Tombstoned, tedy ukončená a dojde k obnovení a načtení stavů ze slovníku stavů.

OnNavigatedTo

Tato metoda je volána vždy, když uživatel přichází na stránku z jiné stránky v rámci běžící aplikace.

Closing

Metoda Closing je vyvolána v okamžiku, když uživatel přejde zpět přes první stránku aplikace. V tomto případě se aplikace ukončí a žádný stav se neukládá. Limit pro provedení ukončení aplikace je 10 sekund, v případě překročení se aplikace ukončí a přeruší vykonávaný kód v rámci metody.

6. Návrh a analýza aplikace

Na začátku kapitoly se budeme věnovat definování funkčních a nefunkčních požadavků na implementovanou aplikaci. V analýze aplikace si popíšeme jednotlivé části aplikace, jak by měly vypadat a co by měly obsahovat.

6.1. Požadavky na aplikaci

Aplikace vychází ze dvou základních proudů požadavků. První proud obsahuje požadavky vyspecifikované v zadání diplomované práce a druhý proud požadavků vznikl při navázání spolupráce s českou komunitou vývojem a správy Jabber serveru v České republice. Zadání diplomové práce definuje spíše obecný požadavek na implementace Jabber klienta a jeho rozšíření o GIS. Česká komunita na druhou stranu vyspecifikovala detailnější požadavky, které odrážely požadavky s ohledem na už existující klienty z vlastního portfolia.

Funkční požadavky:

- Aplikace bude využívat XMPP protokol.
- Aplikace bude používat GPS lokaci mezi uživateli.
- Aplikace bude nabízet sdílení GPS lokace.
- Aplikace umožní spravovat různé uživatelské účty.
- Uživatel bude moci upravovat kontakty.
- Aplikace poskytne nastavení různých statusů.
- Aplikace bude ukládat historii konverzací.

Nefunkční požadavky:

- Uživatelské rozhraní bude v anglickém jazyce.
- Aplikace bude lehce ovladatelná a intuitivní.
- Aplikace bude podporovat operační systém Windows Phone ve verzi 7 a 8.

6.2. Analýza aplikace

Z funkčních požadavků jsme schopni sestavit základní funkcionalitu aplikace. Aplikace by měla být uživatelsky přívětivá a umožňovat uživatelsky intuitivní ovládání. Další ze základ-

ních pilířů by mělo být použití standardů uživatelského rozhrání stanovené Microsoftem pro lepší ovládání. To znamená dodržení velikosti položek pro ovládání a dalších doporučení, které vydala společnost Microsoft pro aplikace Windows Phone.

Implementovaná aplikace bude sloužit jako mobilní Jabber klient. Umožní autentizaci a autorizaci uživatele vůči serveru pomocí uživatelského jména (JID) a hesla. Aplikace si bude moci pamatovat uživatelské účty, které někdy byly použity. Po přihlášení nabídne přehled všech kontaktů (rosters). Kontakty bude třídit podle jejich dostupnosti, to znamená na kontakty, které jsou připojené a odpojené. Kontakty musí jít spravovat formou přidávání nových kontaktů nebo jejich mazání. Hlavní obrazovka musí nabízet vedle přehledu uživatelů taky historii konverzací.

Aplikace musí obsahovat nastavení, kde bude možnost upravovat aktuální chování aplikace a nabízet informaci o verzi aplikace. Jedno z chování aplikace bude vypínání a zapínání sdílení GPS lokace nebo zapnutí a vypnutí zobrazování nepřipojených kontaktů. V případě zapnuté GPS lokace aplikace umožní využití souřadnic k zobrazení aktuální polohy na mapě a pomocí GIS rozšíření filtrovat uživatele na základě vzdálenosti od aktuální polohy.

6.2.1. Přihlašovací obrazovka

Po spuštění aplikace se vždy zobrazí přihlašovací obrazovka, která bude obsahovat pole pro zadání uživatelského jména a hesla pro přihlášení k serveru. Uživatel se přihlásí pomocí tlačítka přihlásit. Přihlašovací obrazovka umožní registrování nového uživatele pomocí odkazu, který přesměruje uživatele na webovou stránku, která obsahuje registrační formulář. Před přihlášením k serveru musí být uživatel dotázán, zda chce uživatelské údaje uložit pro budoucí použití. V případě kdy budou existovat uložené uživatelské údaje, musí být zpřístupněné tlačítko pro správu uživatelských účtu, které přejde na stránku s přehledem uživatelských účtů.

6.2.2. Přehled uživatelských účtů

Seznam uživatelských účtů nabídne přehled všech uložených uživatelských účtů pro přihlášení. Usnadní to práci při používání více účtů a jejich přepínaní mezi jednotlivými účty bez vyplňování uživatelského jména a hesla. Navíc nabídne přidávání, editaci a mazání účtů.

6.2.3. Detail uživatelského účtu

Uživatelský účet bude definován položkami název účtu, uživatelské jméno, heslo, doména serveru, adresa serveru a portem. Všechny položky budou pro uživatelé nastavitelné. Uživatel může nastavit položky přímo vyplněním nebo se položky automaticky nastaví při přihlášení a převzetí položek z JID, které se používá pro přihlášení uživatele. Při manuálním vyplnění položek může uživatel změnit výchozí port a adresu serveru.

6.2.4. Přehled kontaktů

Přihlášený uživatel po přihlášení bude přesměrován na centrální stránku, která bude zobrazovat seznam kontaktů skládající se z avatara[9], uživatelského jména, stavu, statusu a aktuální vzdálenosti od aktuální polohy uživatele. Nabídne správu kontaktů, jejich mazání a přidávání. Stránka bude výchozí pro navigaci na ostatní stránky aplikace. Nabídne přesměrování na následující stránky:

- Detail kontaktu.
- Správa statusu uživatele.
- Zobrazení mapy.
- Přehled historie konverzací.
- Konverzace.

6.2.5. Detail kontaktů

Nabídne detail konkrétního kontaktu. Detail bude obsahovat avatara, uživatelské jméno, GPS souřadnice a mapu s aktuální polohou kontaktu. Detail budu sloužit pouze jako přehled informací o kontaktu, nebude nijak editovatelný.

6.2.6. Správa statusu uživatele

Stránka zobrazí seznam všech stavů, mezi kterými uživatel má možnost přepínat (výčet stavů viz kapitola 2.6.2). Navíc mimo nastavování stavů uživatel bude mít na stránce možnost měnit svůj status, který může nastavit v libovolné textové podobě.

[9] Avatar je pojem pro označení ikony nebo fotky uživatele.

6.2.7. Zobrazení mapy

Stránka bude obsahovat mapový podklad pro zobrazení aktuální polohy uživatele a zobrazení kontaktů, které nabídly pro konzumaci svou aktuální polohu. Poloha bude označena na mapě připínáčkem, který bude obsahovat avatara uživatele, uživatelské jméno a vzdálenost od uživatele. Vzdálenost se bude zobrazovat pouze u kontaktů. Mapa by měla umožnit filtrovat kontakty podle aktuální vzdálenosti od uživatele. Také by mapa měla umět nabídnout historii poloh pro konkrétní kontakt.

6.2.8. Přehled historie konverzací

Historie konverzací by měla být velice podobná s přehledem kontaktů. Navíc by se mělo u položek seznamu kontaktů pro které existuje historie konverzace, zobrazovat část obsahu poslední zprávy a čas poslední zprávy.

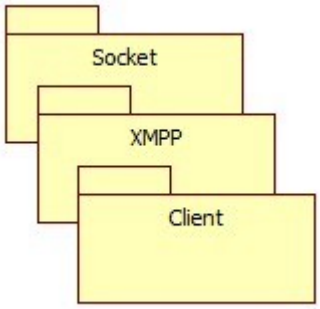
6.2.9. Konverzace

Výměna zpráv mezi uživateli bude probíhat na stránce konverzace. Vždy bude probíhat konverzace s jedním uživatelem, jak to známe při komunikaci pomocí sms[10] zpráv v mobilním telefonu. Jednotlivé zprávy musí jít odlišit podle toho, kdo zprávu odeslal. Zpráva by měla také obsahovat čas odeslání, popřípadě přijetí.

6.3. Návrh

Z popisu funkcionality jednotlivých části aplikace lze vydefinovat entity, které bude potřeba pro implementaci aplikace a jaké vlastnosti budou obsahovat. Aplikace by měla být také rozdělená do třech vrstev. První vrstva by měla být implementovaná jako knihovna, která bude obsluhovat komunikaci. Druhá vrstva bude obsahovat logiku XMPP protokolu a třetí vrstva bude obsahovat aplikační logiku.

[10] SMS označení pro samostatnou krátkou textovou zprávu.



Obrázek 6: Vrstvy aplikace

Socket

Vrstva Socket by měla obsahovat entity, které zajišťují připojení a udržování spojení se vzdáleným serverem. Dále by měla sloužit k zasílání dat a příjmu dat.

XMPP

Obslužná vrstva pro zpracování a identifikaci XMPP zpráv. Vrstva by měla převádět surová data, v našem případě XML zprávy na objekty, které by se dále zpracovávaly na klientské vrstvě. Převádění by mělo probíhat metodou serializace nebo provedení parsování.

Client

Klientská vrstva by měla obsahovat aplikační logiku. Měla by konzumovat objekty vytvořené v XMPP vrstvě a obsahovat uživatelské rozhraní.

7. Popis implementace

Kapitola obsahuje popis nástrojů použitých pro implementaci aplikace, návrhu uživatelského rozhraní a hrnutí postupů jednotlivých kroků implementace.

7.1. Nástroje pro implementaci a návrh aplikace

Aplikace je implementována pomocí vývojového nástroje Microsoft Visual Studio ve verzi 2010 a pomocí Windows Phone SDK ve verzi 7.1.[9] Pro implementaci byl zvolen programovací jazyk C# z platformy .NET Framework. Uživatelské rozhraní aplikace je navrhnuté pomocí nástroje Expression Blend 4. [10]

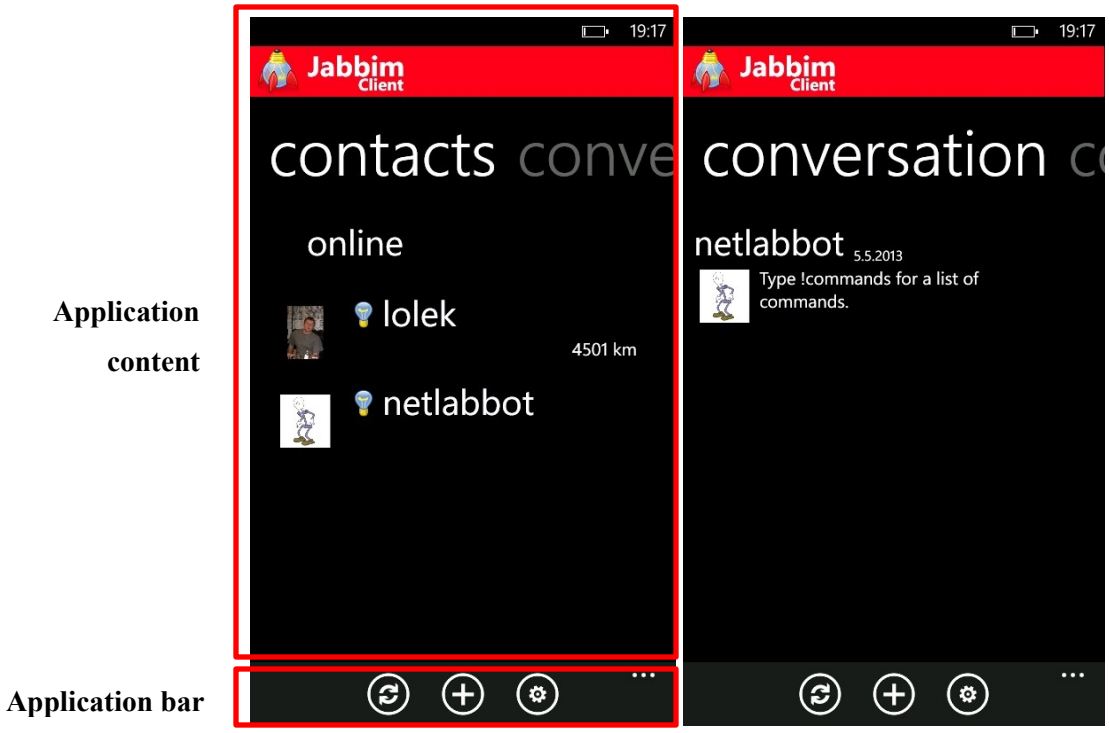
7.2. Uživatelské rozhraní

Návrh uživatelského rozhraní je navrhnuté pomocí nástroje Expression Blend. Tento nástroj umožňuje načrtnout uživatelské prostředí bez programování. To umožní se podívat na aplikaci bez programátorského myšlení a spíše se zaměřit na uživatelské pohodlí. Programátoři také ocení automatické vygenerování struktury aplikace na základě definovaného návrhu stránek.

Celé uživatelské rozhraní aplikace má strukturu odpovídající struktuře standardu Microsoftu. Je rozdělena na dvě části:

- **Application content** je samotný obsah aplikace a zabírá největší prostor aplikace.
- **Application bar** je aplikační lišta, kde se nacházejí dva typy prvků. Buď se jedná o kulatou ikonu s popiskem, nebo o textovou hodnotu.

Centrální stránka [Obrázek 7] s přehledem kontaktů a historie konverzací využívá komponentu s názvem Pivot, která řadí stránky vedle sebe a simuluje efekt navazujících stránek na sebe.



Obrázek 7: Ukázka struktury centrální stránky a použití komponenty Pivot.

Centrální stránka se odlišuje od všech ostatních stránek navrhnutých v aplikaci díky použité komponentě, která umožňuje cyklický přechod mezi kontakty a konverzacemi. Obě stránky mají jeden zdrojový kód, viz ukázka zdrojového kódu níže. Ostatní stránky jsou tvořeny jednoduchou šablonou stránky. Tedy neobsahují žádnou speciální komponentu pro uspořádání stránek vedle sebe.

Ukázka zdrojového kódu:

```
<!--Pivot Control-->
<controls:Pivot Foreground="White" Margin="0,59,0,0">
    <!--Pivot item one-->
    <controls:PivotItem Header="contacts" Margin="12,8,12,20">
        <Grid>
            <LisBox Height="502" HorizontalAlignment="Left" Margin="0,0,0,0"
                    Name="lstRoster" VerticalAlignment="Stretch" Width="457"
                    VerticalContentAlignment="Stretch" ItemsSource="{Binding}"
                    ItemContainerStyle="{StaticResource ListBoxItemStyle}"
                    SelectionChanged="lstRoster_SelectionChanged">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel Orientation="Horizontal" IsHitTestVisible="True">
                        <toolkit:ContextMenuService.ContextMenu>
                            <toolkit:ContextMenu>
                                <toolkit:MenuItem Header="detail"
                                        Name="detailItem" Click="detailItem_Click" />
```

```
                        <toolkit:MenuItem Header="delete"
                                Name="deleteItem"
                                Click="deleteItem_Click" />
                    </toolkit:ContextMenu>
                </toolkit:ContextMenuService.ContextMenu>
                <StackPanel Orientation="Horizontal" >
                    <Image Source="{Binding AvatarImagePath}" Margin="20" Width="60"
                           Height="60"></Image>
                </StackPanel>
                <StackPanel Orientation="Vertical">
                    <StackPanel Orientation="Horizontal">
                        <Image Source="{Binding StatusImagePath}" Width="32"
                               Height="32" Margin="0,5,5,0"></Image>
                        <TextBlock Text="{Binding Username}" FontFamily="Segoe WP"
                                   FontSize="40" Style="{Binding RosterStyle}">
                        </TextBlock>
                    </StackPanel>
                    <StackPanel Orientation="Horizontal">
                        <Grid>
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="250"/>
                                <ColumnDefinition Width="*"/>
                            </Grid.ColumnDefinitions>
                            <TextBlock Grid.Column="0" Text="{Binding Status}"
                                       FontFamily="Segoe WP" FontSize="20"
                                       HorizontalAlignment="Left">
                            </TextBlock>
                            <TextBlock Grid.Column="1"
                                       Text="{Binding DistanceLocation}"
                                       FontFamily="Segoe WP" FontSize="20"
                                       HorizontalAlignment="Right"
                                       TextAlignment="Right">
                            </TextBlock>
                        </Grid>
                    </StackPanel>
                </StackPanel>
            </StackPanel>
            </DataTemplate>
        </ListBox.ItemTemplate>
        </ListBox>
    </Grid>
</controls:PivotItem>
<!--Pivot item two-->
.....
```

Application bar obsahuje v celé aplikaci maximálně tři ikony, což opět odráží doporučený počet ikon Microsoftem pro návrh Windows Phone aplikací, který je stanoven na maximální počet pět.

Obrázek 8: Ukázka přihlašovací stránky a správy statusu.

7.3. Implementace testovací aplikace

Pro použití aplikace na platformě Windows Phone byla potřeba v prvním kroku zjistit, zda je možnost použít knihovnu pro používání XMPP protokolu, tedy jestli existuje nějaká použitelná knihovna třetí strany pro zvolenou platformu nebo bude potřeba implementovat vlastní knihovnu. XMPP Standart Foudation[11] nabízí na svých stránkách seznam knihoven pro implementaci vlastních Jabber klientů.

Navrhovaná aplikace je omezena platformou. Jelikož se jedná o relativně novou platformu, byla nalezena pouze jedna knihovna odpovídající všem požadavkům. Společnost AG Software[12] nabízí SDK s názvem MatriX XMPP pro Windows Phone a umožňuje používat i řadu již implementovaných rozšíření protokolu. Proto byla tato knihovna zvolena pro použití v navrhované aplikaci a z důvodu lehké použitelnosti k implementaci první testovací aplikace, která by prověřila, zda zvládá základní komunikaci se servery a také prověří propustnost rozšíře-

[11] XMPP Standart Foudation je nezisková společnost, která se stará o správu standartu XMPP protokolu.
[12] http://www.ag-software.net

ní XEP-0080[13] u různých serverů, protože jak jsme si už řekli dříve, servery jsou také rozlišeny nabízenými službami. Díky funkčního požadavku bude aplikace používat GPS lokaci mezi uživateli, proto je tento test propustnosti velice důležitý.

7.3.1. MatriX XMPP SDK

Pro otestování všech potřebných funkcionalit byla použita trial verze knihovny. Celé SDK je distribuováno jako instalační balíček exe. Trial verze neobsahovala žádné omezení ve funkčnosti a nebyla ani omezena časovým intervalem použití. Pro přechod na plnou verzi stačí vložení licenčního kódu, jako vlastnost komponenty. Jediný rozdíl trial verze od plnohodnotného produktu byla existence informační zprávy při každé odeslané zprávy.

Instalace samotného balíčku nenabízí žádné možnosti modifikace instalace, kromě volby umístění instalovaný souborů. Instalace provede do zvoleného umístění nakopírování potřebných souborů a knihoven dll[14]. Důležitou nakopírovanou knihovnou je soubor s názvem Matrix.dll. Tento soubor je potřeba vložit jako referenci do vlastního projektu ve Visual Studiu. Popisu přidání reference ve Visual Studiu se v této diplomové práci věnovat nebudu.

7.3.2. Testovací aplikace

Testovací aplikace byla založena jako Windows Phone aplikace bez použití jakékoliv šablony. Pro test byla vytvořena jedna stránka, která byla pro celý projekt hlavní. Projekt měl referenci na potřebnou knihovnu Matrix. Z knihovny byl potřeba použít objekt XMPPClient, který byl přidán jako privátní proměnná na hlavní stránce a jeho nová instance se vytváří v konstruktoru stránky.

```
// Constructor
public MainPage()
{
    InitializeComponent();

    if (_xmppClient == null)
        _xmppClient = new XmppClient { XmppDomain = "jabbim.cz",
            Username = "DiplomkaTest", Password = "diplomka" };
}
// XMPP Client
private XmppClient _xmppClient;
```

[13] XEP-0080 – User Location: http://xmpp.org/extensions/xep-0080.html
[14] Dll je dynamicky linkovaná knihovna, která splňuje koncept sdílených knihoven stanovený společnosti Microsoft.

XmppClient pro připojení na server potřebuje mít vyplněné atributy XmppDomain, Username a Password. XmppDomain obsahuje doménu serveru. Username a Password jsou uživatelské jméno a heslo, které slouží k přihlášení uživatele na server. Pokud jsou všechny atributy dobře vyplněny lze se připojit na server pomocí procedury Open.

```
private void btnConnect_Click(object sender, RoutedEventArgs e)
{
    try
    {
        _xmppClient.Open();
    }
    catch (Exception e)
    {
        MessageBox.Show(String.Format("Error: {0}", e.Message));
    }
}
```

Aplikace obsahuje tři tlačítka, která spouštějí různé akce. Z předchozí ukázky kódu, jde vidět, že přihlášení uživatele se provádí po stisku tlačítka btnConnect. Další dvě tlačítka slouží pro test odeslání zprávy typu Chat a pro odeslání polohy uživatele bez použití GPS přijímače v přístroji. Odeslat zprávu lze vytvořením nové instance objektu typu Message a vložením jako parametru do metody Send(), viz kód níže.

```
private void btnSendMessage_Click(object sender, RoutedEventArgs e)
{
    // basic send message example
    Message message = new Message();
    message.To = "TestSend@jabbim.cz";
    message.Type = MessageType.chat;
    message.Body = "Hello World";

    try
    {
    _xmppClient.Send(message);
    }
    catch (Exception e)
    {
        MessageBox.Show(String.Format("Error: {0}", e.Message));
    }
}
```

Posledním důležitým testem funkcionality bylo odeslání polohy uživatele, tedy byla potřeba otestovat rozšíření protokolu s označením XEP-0080. Zde nastává první problém, protože knihovna toto rozšíření nepodporuje. Problém byl vyřešen tím, že knihovna podporovala posílání XML zprávy na server. Proto stačilo vytvořit XML obsah reprezentující zprávu odpovídající standartu XEP-0080 pro posílání uživatelské polohy.[5]

```
private void btnPublishLocation_Click(object sender, RoutedEventArgs e)
{
    string xml = "<message from=\"DiplomkaTest@jabbim.cz\"
                          to=\"TestSend@jabbim.cz\">
                    <event xmlns=\"http://jabber.org/protocol/pubsub#event\">
                      <items node=\"http://jabber.org/protocol/geoloc\">
                        <item id=\"d81a52b8-0f9c-11dc-9bc8-001143d5d5db\">
                          <geoloc xmlns=\"http://jabber.org/protocol/geoloc\"
                                  xml:lang=\"en\">
                            <accuracy>10</accuracy>
                            <lat>10</lat>
                            <locality>Venice</locality>
                            <lon>10</lon>
                          </geoloc>
                        </item>
                      </items>
                    </event>
                  </message>";
    XmppXElement xmlMessage = XmppXElement.LoadXml(xml);
    _xmppClient.Send(xmlMessage);
}
```

Všechny testy byly provedeny na několika známých serverech, které mi poskytla česká komunita Jabber.

Testované servery:

- Jabbim.cz
- Jabber.cz
- Jabber.org
- Chat.facebook.com

Test vždy probíhal mezi mobilním zařízením, na kterém běžela testovací aplikace a desktopovým klientem. Testovací aplikace se přihlašovala vždy stejným uživatelským jménem. Desktopový klient používal při každém testu rozdílné přihlašovací údaje pro různé servery. Komunikace mezi klienty byla sledována pomocí desktopového klienta, který obsahoval logovací konzoli všech příchozích a odchozích zpráv. Všechny testy byly provedeny úspěšně. Došlo k ověření jak správné funkcionalitě knihovny, tak také propustnosti serverů.

Knihovna nakonec nebyla použita pro implementaci mobilního XMPP klienta z důvodu vysoké pořizovací ceny za licenci.

7.4. Implementace klienta

Pro samotnou implementaci nebyla použita knihovna MatriX z důvodu vysoké ceny licence. Proto po konzultaci s českou komunitou Jabber bylo doporučeno použití knihovny xMedianet[15], která se nacházela ve verzi Beta testů a nabízela základní implementaci komunikace XMPP protokolu.

XMedianet knihovna je distribuována ve formě zdrojových kódů pod volno licenci k šíření. Nevýhoda této komponenty je její neotestovaná funkcionalita, neexistující dokumentace a její poslední aktualizace z 6. dubna 2012. Tento stav ukazuje na neprobíhající vývoj, proto také nebyla zveřejněna jako oficiální knihovna pro implementaci XMPP klientů. Na druhou stranu se jedná o volně šiřitelný zdrojový kód, což umožňuje rozšiřování stávající funkcionality, popřípadě opravy chyb. Také se jedná použitelný základ pro implementaci nového klienta.

Zdrojové kódy byly dostupné ke stažení jako sada projektů ve Visual Studiu a obsahovaly mnoho testovacích projektů. Ze všech projektů byly vybrané dva projekty. První s názvem socket, který provádí obsluhu TCP spojení pomocí socketu. Druhý s názvem xmpp, který obsahuje logiku XMPP protokolu. Oba projekty byly implementované jako projekty typu library a psány v jazyce C#.

Díky změně knihovny bylo potřeba opět provést otestování základní funkcionality knihovny, jestli odpovídá všem základním požadavkům. Testování v tomto případě nemělo za následek vznik testovací aplikace, která by simulovala základní požadavky. V případě, že by vznikl problém nebo knihovna by neobsahovala požadovanou funkcionalitu, byla by doimplementovaná díky dostupnosti zdrojových kódů. Jako základ pro vznik mobilní klienta vznikl projekt XMPPClient.

7.4.1. XMPPClient

XMPPClient tvoří projekt typu Windows Phone application. Jedná se teda o cílovou aplikaci klienta. Klient byl vytvořen s vygenerovanými stránkami aplikace díky návrhu celého uživatelského rozhraní v nástroji Expression Blend, který vygeneroval jednotlivé stránky podle návrhu.

[15] http://xmedianet.codeplex.com/

Aplikace je definovaná hlavní třídou App, která obsahuje definici globálních stylů a proměněných pro použití napříč aplikací. Základním objektem v této třídě je XMPPPhoneClient, který je typu XMPPClient. Tento objekt reprezentuje klienta a řeší logiku XMPP protokolu.

Objekt XMPPPhoneClient je v aplikaci použit jako singleton, to znamená, že v celé aplikaci existuje pouze jedna instance tohoto objektu. Singleton neobsahuje ani tzv double check pro více vláknové aplikace [8], protože v tomto případě se k tomuto objektu nepřistupuje z více vlákem viz kód níže.

```
//Client library
private static  XMPPClient xMPPPhoneClient;
public static XMPPClient XMPPPhoneClient
{
    get
    {
        if (xMPPPhoneClient == null)
        {
            xMPPPhoneClient = new XMPPClient();
        }
        return xMPPPhoneClient;
    }
}
```

V aplikaci jsou použity především následující vlastnosti a procedury tohoto objektu.

Vlastnosti:

- XMPPPhoneClient.Server – adresa serveru.
- XMPPPhoneClient.Port – port, na kterém běží serverová aplikace.
- XMPPPhoneClient.Resources – nastavení resources pro klienta.
- XMPPPhoneClient.XMPPAccount – objekt, který obsahuje informace o uživatelském účtu.
- XMPPPhoneClient.AutoReconnect – nastavení znovu připojení při výpadku spojení.
- XMPPPhoneClient.Connected – indikace zda je klient připojen k serveru.
- XMPPPhoneClient.PresenceStatus – informace o aktuálním stavu a statusu uživatele.
- XMPPPhoneClient.RosterItems – kolekce kontaktů uživatele.

Procedury:

- XMPPPhoneClient.Connect() – přihlášení klienta k serveru.
- XMPPPhoneClient.Disconnect() – odpojení klienta od serveru.

- XMPPPhoneClient.SendChatMessage(string strMessage, JID jidto) – odeslání textové zprávy typu chat určenému kontaktu.
- XMPPPhoneClient.SetGeoLocation(double flat, double flot) – nastavení polohy uživatele a odeslání všem kontaktům.

Pro příjem zpráv a informací o kontaktech se v aplikaci registrují následující obslužné EventHandlery:

- XMPPPhoneClient.OnNewConversationItem – vyvolá se tehdy, pokud klient přijme textovou zprávu od jakéhokoliv kontaktu.
- XMPPPhoneClient.OnRosterItemsChanged – vyvolá se při změně existujícího kontaktu uživatele.

7.4.2. Získávání polohy

Získání geografické polohy se provádí použitím namespace Location, který obsahuje potřebné objekty pro získání aktuální polohy. Pro získání polohy slouží objekt typu GeoCoordinateWatcher.[12] Při inicializaci objektu se nastavuje, jak vysokou přesnost polohy požadujeme. Existují dvě možnosti nastavení:

- Default – vrací polohu na základě dostupných prostředků, tak aby stálo co nejméně energie. Proto v případě použití polohy z BTS nemusí být poloha přesná.
- High – vrací nepřesnější polohu bez ohledu na dostupných prostředcích. Bude použita GPS místo méně přesných dat z BTS a WiFi.

Další možností nastavení objektu GeoCoordinateWatcher je MovementThreshold, který omezuje změny polohy o malé změny. To znamená, že bude ignorovat změny polohy do nastavené hodnoty. V případě, že nastavíme tuto vlastnost na hodnotu 100, dojde ke změně polohy nejdříve po 100 metrech od předchozí zaznamenané polohy. V aplikaci je použita třídu Settings, která odráží nastavení uživatele. Nastavení si uživatel může měnit, zda chce používat zjišťování polohy nebo kolik metrů chce ignorovat pro změnu polohy.

```
//GPS watcher
private GeoCoordinateWatcher watcher;

public void StartLocation()
{
    if (Settings.AutomaticSetposition)
    {
        if (watcher == null)
        {
            watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.High);
            watcher.MovementThreshold =
                 Settings.MovementsDistance;
            watcher.PositionChanged += new EventHandler
               <GeoPositionChangedEventArgs<GeoCoordinate>>
               (watcher_PositionChanged);
        }
        watcher.Start();
    }
}
```

Pro zpracování získané polohy je potřeba zaregistrovat event handler PositionChanged, který se vyvolá při změně polohy při splnění definovaných podmínek. V případě vyvolání se provede metoda, která nastaví aktuální polohu a odešle změnu všem kontaktům, které jsou dostupné, viz následující kód.

```
private void watcher_PositionChanged
(object sender, GeoPositionChangedEventArgs<GeoCoordinate> e)
{
    if ((e.Position.Location.Latitude != CurrentLocation.lat)
       || (e.Position.Location.Longitude != CurrentLocation.lon))
    {
        CurrentLocation.lat = e.Position.Location.Latitude;
        CurrentLocation.lon = e.Position.Location.Longitude;
    }

    if (watcher.Status == GeoPositionStatus.Ready
       && !watcher.Position.Location.IsUnknown
       && Settings.AutomaticSetposition)
    {
        XMPPPhoneClient.SetGeoLocation
        (watcher.Position.Location.Latitude, watcher.Position.Location.Longitude);
    }
}
```

K výpočtu vzdálenosti kontaktu od uživatele a k filtrování kontaktů podle vzdálenosti se používá třída GeoLocationUtils, která obsahuje dvě funkce pro přepočet vzdálenosti mezi dvěma polohama. Jedna funkce vrací vzdálenost v kilometrech a druhá v metrech.

```
public static Double DistanceInMetres(double lat1, double lon1, double lat2, double lon2)
        {

            if (lat1 == lat2 && lon1 == lon2)
                return 0.0;
            var theta = lon1 - lon2;

            var distance = Math.Sin(deg2rad(lat1)) * Math.Sin(deg2rad(lat2)) +
                           Math.Cos(deg2rad(lat1)) * Math.Cos(deg2rad(lat2)) *
                           Math.Cos(deg2rad(theta));

            distance = Math.Acos(distance);
            if (double.IsNaN(distance))
                return 0.0;

            distance = rad2deg(distance);
            distance = distance * 60.0 * 1.1515 * 1609.344;

            return (distance);
        }
```

7.4.3. Ukládání dat

Aplikace si ukládá uživatelské účty, historii konverzací a nastavení aplikace. Pro každý uložený uživatelský účet se ukládá nastavení aplikace a historie konverzací. Z důvodu malé rozsáhlosti ukládaných dat byla pro ukládání zvolena metoda serializace dat do formátu XML. To znamená, že aplikace si uloží předdefinované objekty jako XML soubor do uložiště, v našem případě do Isolated storage aplikace.[13]

Ukládané objekty:

- Kolekce účtů – obsahuje všechny uložené uživatelské účty.
- Kolekce historických konverzací – pro jeden uživatelský účet se ukládá celá historie konverzací omezena na posledních 20.
- Nastavení aplikace – pro každý uživatelský účet se ukládá individuální nastavení aplikace.

Uložení a načtení dat probíhá pomocí třídy SerializationHelper, který obsahuje proceduru Save a funkci Load.

Procedura Save se volá dvěma atributy, první reprezentuje ukládány objekt a druhý název souboru, do kterého se objekt uloží.

```
public void Save(object objectForSerialization, string fileName)
{
    IsolatedStorageFile storage =
        IsolatedStorageFile.GetUserStoreForApplication();
    IsolatedStorageFileStream stream = null;
    try
    {
        stream = storage.CreateFile(fileName);
        XmlSerializer xml = new XmlSerializer(objectForSerialization.GetType());
        xml.Serialize(stream, objectForSerialization);
    }
    catch (Exception ex)
    { }
    finally
    {
        if (stream != null)
        {
            stream.Close();
            stream.Dispose();
        }
    }
}
```

Funkce Load se také volá s dvěma atributy, první obsahuje název souboru a druhý typ objektu, který chceme ze souboru načíst. Funkce vrací samotný serializovaný objekt, který odpovídá požadovanému typu.

```
public static object Load(string fileName, Type type)
{
    IsolatedStorageFile storage = IsolatedStorageFile.GetUserStoreForApplication();
    object tmpObject;
    if (storage.FileExists(fileName))
    {
        IsolatedStorageFileStream stream = null;
        try
        {
            stream = storage.OpenFile(fileName, FileMode.Open);
            XmlSerializer xml = new XmlSerializer(type);
            tmpObject = xml.Deserialize(stream);
        }
        catch (Exception ex)
        {
            tmpObject = null;
        }
        finally
        {
            if (stream != null)
            {
                stream.Close();
                stream.Dispose();
            }
        }
    }
    else
        tmpObject = null;
    return tmpObject;
}
```

8. Publikace mobilní aplikace na Marketplace

V kapitole si přiblížíme základní premisy pro úspěšné publikování vlastní aplikace na Marketplace.

8.1. Windows Phone Marketplace

Jedná se o virtuální obchod společnosti Microsoft. Slouží pro distribuci všech aplikací pro Windows Phone platformu. Vývojáři zde publikují své vlastní aplikace, které si uživatelé mohou stáhnout. Všechny aplikace jsou děleny do skupin podle toho, k čemu jsou jednotlivé aplikace používány. Aplikace mohou být zpoplatněné nebo být zcela zdarma.

Aplikace v Marketplace jsou identifikované názvem, ikonou a popisem. Každá aplikace může obsahovat recenze uživatelů, což umožní rozhodování uživatelům mezi jednotlivými aplikacemi a uživatel má možnost si udělat vlastní názor o funkčnosti aplikace. Na základě recenzí od uživatelů je aplikace ohodnocena celkovým hodnocením, které je reprezentováno počtem získaných hvězdiček od jedné do pěti, kdy pět je nejlepší hodnocení. Uživatel má také možnost si prohlédnout jak aplikace vypadá díky přiložených screenshotů aplikace.

8.2. Premisy pro publikování vlastní aplikace

Pokud disponujete implementovanou aplikací, kterou chcete zveřejnit, je potřeba být registrovaný na stránce Windows Phone Dev Centre[16].[11] Pro přihlášení na stránku si vystačíte také s Live ID. Stránka shrnuje všechny potřebné informace o publikování aplikace. Uživatel pro publikování aplikace musí být registrovaný jako vývojář, kterým se stane uhrazením ročního poplatku 99$. Výjimku tvoří uživatelé, kteří jsou registrování studenti pomoci programu Dreamspark[17].

Úspěšnému publikování aplikace předchází schvalovací proces ze strany Microsoftu, který provádí kontrolu, zda aplikace splňuje všechny podmínky pro udělení certifikátu. Certifikace určuje, zda aplikace může být zveřejněna v Marketplace. Certifikační proces lze do značné míry nasimulovat ve vývojovém prostředí Visual Studio. Vývojové prostředí obsahuje nástroj

[16] https://dev.windowsphone.com/en-us
[17] https://www.dreamspark.com/

s názvem Marketplace Test Kit. Tento nástroj otestuje aplikaci na různé scénáře pomocí jednotlivých testů.

Automatický test

Provádí validaci soubor XAP[18], který reprezentuje instalační soubor aplikace. Test zkontroluje velikost a obsah souboru, zda je validní pro instalaci. Dále test provede validaci vlastností aplikace. Prověří, jaké všechny zdroje zařízení aplikace bude využívat. Mezi zdroje patří datové připojení, GPS přijímač, kontakty, fotoaparát atd. Poslední část testu provede validace ikon a screenshotů, zda mají správný formát.

Monitorovaný test

Test monitoruje běh spuštěné aplikace, kterou má uživatel spuštěnou a simuluje práci uživatel. Test na základě monitorování aplikace vyhodnotí následující testovací scénáře:

- Časový interval spuštění aplikace.
- Maximální použití paměti přístroje.
- Zaznamenání neočekávaného ukončení aplikace.
- Správné použití tlačítka back.

Manuální test

Obsahuje 50 testovacích scenářů, které si uživatel manuálně otestuje a zaeviduje, zda je aplikace splňuje. V podstatě se jedná o formulář s jednotlivými scénáři a evidenci výsledku testu.

Provedení všech testů nezaručuje, že aplikace bude automaticky připravena a schválená pro zveřejnění v Marketplace. Testy slouží pouze pro usnadnění přípravy pro oficiální proces certifikace. Oficiální schvalovací proces ověřuje, zda aplikace splňuje požadavky, které se dělí na následující skupiny [14]:

- Aplikační politika Windows Phone.
- Obsahová politika Windows Phone.

[18] XAP je formát souboru, který se používá pro distribuci a instalaci Windows Phone aplikací.

- Validace instalačního balíčku.
- Kontrola technických vlastností aplikace.

Při odeslání aplikace do certifikačního procesu je důležité mít připravené nejenom instalační XAP soubor aplikace, ale je potřeba mít připravené další povinné položky, které se zadávají při odesílání aplikace. Seznam povinných položek obsahuje:

- Ikony aplikace v rozlišení 99 x 99, 173 x 173 a 200 x 200 px, bez průhledného pozadí – tyto ikony jsou zobrazeny v nabídce Marketplace.
- Název aplikace a její popis, který bude zobrazen v Markeptlace.
- Kategorie aplikace a klíčová slova pro vyhledávání.
- Několik nasnímaných obrazovek aplikace v rozlišení 480 x 800 px – maximálně 8 obrázků.
- Ujasnit si cenovou politiku prodeje aplikace.
- Mít připravenou případné poznámky pro certifikačního pracovníka aplikace, tehdy kdy jsou potřeba, například přihlašovací údaje k otestování aplikace.

Celý certifikační proces se vyhodnocuje několik dnů, záleží na vytíženosti certifikačního střediska. Po úspěšné certifikaci je aplikace do 24 hodin publikována na Marketplace.

9. Závěr

Během zpracování této diplomové práce jsem nastudoval problematiku protokolu XMPP a využití tohoto protokolu v Jabber síti. Dále jsem se seznámil s operačním systémem Windows Phone a jeho nástroji pro vývoj vlastní aplikace. Na základě definovaných cílů práce byla provedena analýza a implementace Jabber klienta na platformě Windows Phone. Klient byl implementován pomocí knihovny xMedianet, která obsahovala základní logiku komunikace protokolu XMPP. Knihovna byla doporučena českou komunitou zabývající se vývojem a problematikou Jabber sítí. Jelikož nesplňovala všechny požadavky stanovené v cílech diplomové práce, byla knihovna modifikována o implementaci rozšíření protokolu XEP-0080, který definuje standard pro výměnu uživatelské polohy pomoci XMPP protokolu.

Implementovaná aplikace byla testovaná ve spolupráci s českou komunitou Jabber. Kvůli absenci mobilních přístrojů s operačním systémem Windows Phone testy probíhali vždy mezi implementovanou aplikací na mobilním zařízení a desktopovým klientem. Komunikace byla monitorována na desktopovém klientovi, který obsahoval konzoli pro evidenci všech příchozích a odchozích zpráv. Na základě sledování komunikace byla provedena kontrola správnosti přijatých zpráv implementovaného klienta. Testy probíhaly pod různými uživatelskými účty registrovanými u různých serverů. V průběhu provádění testů se vyskytly problémy se stabilitou aplikace, při načítání velkého počtu kontaktů došlo k pádu aplikace. Tento problém byl opraven optimalizací načítání kontaktů uživatele.

Otestovaná aplikace podstoupila schvalovací procesy pro distribuci na Windows Phone Marketlace. Aplikace byla ke schválení zaslána celkem dvakrát. Při prvním schvalování byla aplikace zamítnuta z důvodu porušení aplikační politiky ve dvou požadavcích. Nebyl splněn mechanismus ověření, zda je uživatel starší třináct let v případě, kdy aplikace umožňuje komunikaci mezi uživateli prostřednictvím výměny zpráv mezi sebou. Dále nebyl splněn požadavek, který stanovuje povinnost informovat uživatele o způsobu zpracování dat obsahující jeho polohu. Tyto nedostatky byly odstraněny a v případě druhého schvalovacího procesu aplikace získala certifikaci pro distribuci v Marketplace.

Celá diplomová práce mi přinesla bohaté zkušenosti týkající se problematiky operačního systému Windows Phone. Doufám, že tyto nabité zkušenosti se budou i nadále zdokonalovat

díky navázané spolupráce s českou komunitou Jabber a implementována aplikace bude do budoucna rozšiřována o nové funkcionality, které budou komunitou vyžadovány. Všechny tyto skutečnosti by měly zajistit to, že aplikace tzv. „neusne na vavřínech“ a nestane se šuplíkovou záležitostí.

Literatura

[1] Peter Saint-Andre. *XMPP: The Definition Guide: Building Real-Time Applications with Jabber Technologies.* O'Reilly Media; 1 edition; 2009, ISBN 978-096521264.

[2] D. J. Adams. *Programming Jabber: Extending XML Messaging.* O'Reilly Media; 1 edition; 2002, ISBN 978-0596002022.

[3] Jabber Software Foudation. *Extensible Messaging and Presence Protokol (XMPP) : Core.* [online]; 2004, Dostupné z: < http://xmpp.org/rfcs/rfc3920.html>.

[4] Jabber Software Foudation. *Extensible Messaging and Presence Protokol (XMPP) : Instant Messaging and Presence.* [online]; 2011, Dostupné z: < http://xmpp.org/rfcs/rfc6121.html>.

[5] Hildebrand Joe. *XEP-0080: User Location.* [online]; 2009, Dostupné z: < http://xmpp.org/extensions/xep-0080.pdf>.

[6] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. *Extensible Markup Language (XML) 1.0.* [online]; 1998, Dostupné z < http://www.w3.org/TR/REC-xml/>.

[7] Tuček Jan. *Geografické informační systémy – Principy a praxe.* Computer Press, 1998, ISBN 80-7226-091-X

[8] Petzold Charles. *Programming Windows Phone 7.* WA: Microsoft Press; 2010, ISBN 978-073-5643-352.

[9] MSDN Library. *Windows Phone SDK Tools.* [online]; 2013, Dostupné z < http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402523(v=vs.105).aspx>

[10] MSDN Library. *How to create your first app for Windows Phone.* [online]; 2013, Dostupné z < http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402526(v=vs.105).aspx>

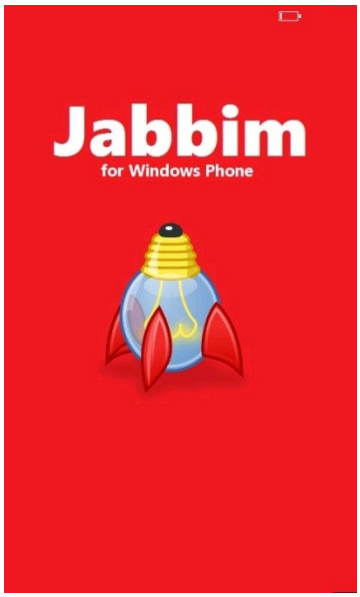
[11] MSDN Library. *How to register your phone for development.* [online]; 2013, Dostupné z < http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff769508(v=vs.105).aspx >

[12] MSDN Library. *How to test apps that use location data for Windows Phone.* [online]; 2013, Dostupné z <http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402526(v=vs.105).aspx>

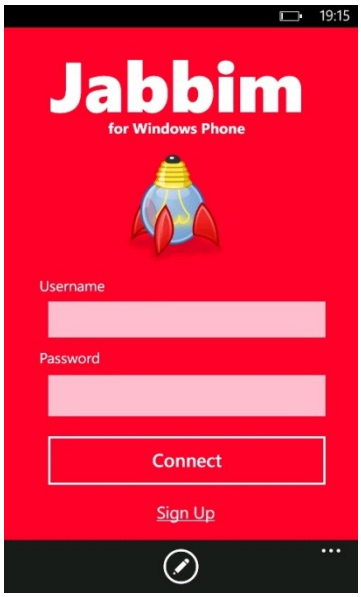
[13] MSDN Library. *Introducing XML Serialization.* [online]; 2013, Dostupné z < http://msdn.microsoft.com/en-US/library/182eeyhh(v=vs.100).aspx >

[14] MSDN Library. *App certification requirements for Windows Phone.* [online]; 2013, Dostupné z <http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184843(v=vs.105).aspx>

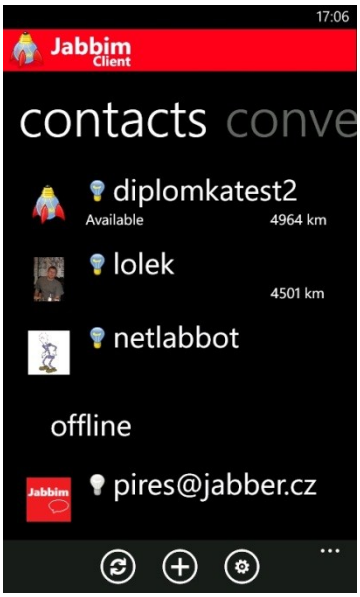
A. Příloha obrázky



Obrázek A.1: Úvodní stránka.

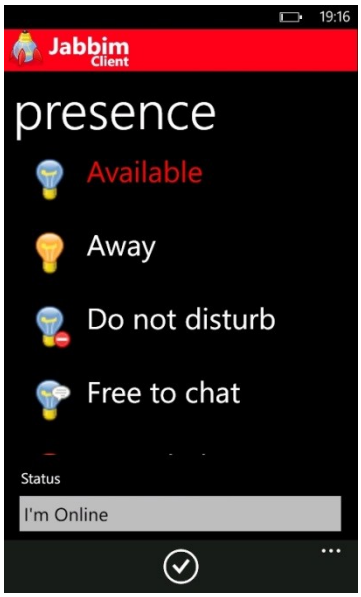


Obrázek A.2: Přihlašovací stránka.



Obrázek A.3: Přehled kontaktů.

Obrázek A.4: Konverzace.



Obrázek A.5: Správa statusu.

Obrázek A.6: Poloha kontaktů na mapě.

B. Příloha na CD/DVD

Obsah přiloženého CD

- src – Zdrojové kódy aplikace